International Journal of Mathematics Trends and Technology
ISSN: 2231 – 5373 /doi:10.14445/22315373/IJMTT-V67I6P504
Volume 67 Issue 6, 35-37, June, 2021

A New Efficient Quantum Algorithm for Computing Primality Measure

Gaurav Saxena[*1], Shilpee Srivastava Saxena[*2]

[1]*Department of Computer Science,* [2]*Department of General Sciences*
BITS Pilani, Dubai Campus, UAE (Former)

Abstract: *In this article we introduce a new concept of Primality Measure based on repeated application of Euler Totient Function. Also, we introduce a new Quantum Algorithm to compute this Primality measure based on Classical Algorithm for Euler Totient Function and Quantum Algorithm for factorization of a number. We next prove that the Classical and Quantum Algorithm used in our New Algorithm (Ramanil Algorithm) are both polynomial time & hence Ramanil Algorithm is also a polynomial time algorithm.*

Keywords: *Euler Totient Function, Primality Measure, Quantum Algorithms, Classical Algorithms, Polynomial Time Complexity.*

I. INTRODUCTION

We propose a new Primality Measure (PM) which is based on repeated application of Euler Totient Function. To compute PM we compute two numbers Num1 & Num2 defined as follows.

Num1 is the number of times one needs to take repeated Totient of a number to reach 1.

For example, $\text{Totient}\,(12) = 4$, $\text{Totient}\,(4) = 2$ & $\text{Totient}\,(2) = 1$.

So therefore, $\text{Num1}(12) = 3$, since the repeated totient needs to be applied 3 times to reach 1.

Num2 is the sum of Num1 on all the divisors of the number.

For example,

$$\text{Num2}(12) = \text{Num1}(12) + \text{Num1}(6) + \text{Num1}(4) + \text{Num1}(3) + \text{Num1}(2) + \text{Num1}(1)$$
$$= 3 + 2 + 2 + 2 + 1 + 0 = 10.$$

Then PM(12) = $\frac{\text{Num1}(12)}{\text{Num2}(12)} = \frac{3}{10}$.

The Primality Measure defined here is the ratio of Num1 & Num2. Thus, Primality Measure of 12 is $3/10$ & therefore 12 is 30% prime. It can be easily seen that all primes are 100% prime by this new measure. This can be proved as follows:

It is obvious that

$$\text{Num1 (prime)} = \text{Num2 (prime)}$$

Since, Num2 (prime) is a sum of Num1 on all divisors of prime. The only divisors of prime are itself and 1 (Num1 of 1 is zero).
Hence Num1 (prime) = Num2 (prime) and therefore PM (prime) = 1 and 100%.

II. ALGORITHM: (RAMANIL ALGORITHM)

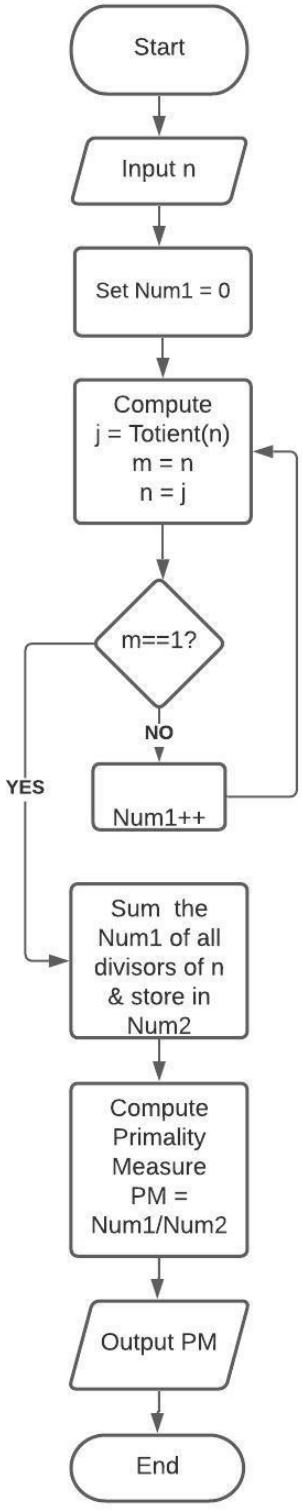

Fig 1. Flowchart of Ramanil algorithm

Before describing the new algorithm, we would like to mention that polynomial time means polynomial function of $log\ (n)$. Since the number n is stored in computer's memory in $log\ n$ bits. We give below the step wise description of our Ramanil Algorithm:

Step 1: Call procedure Num1 = ComputeNum1(n)

Step 2: Call procedure Num2 = ComputeNum2(n)
Step 3: PrimalityMeasure(n) = Num1/Num2
Step 4: Return PrimalityMeasure(n)

The step wise description of ComputeNum1(n) procedure is as follows:

Step1: Set i = 0
Step2: Repeat Step3 till Totient(n) = n
Step 3: Compute j = Totient(n) & set m = n & set n = Totient(n) & increment i by 1. Go to step 2
Step 4: Return i.(This is the value of Num1)

Next, we give the steps wise description of ComputeNum2 procedure:

Step1: Compute the set of divisors of n using Quantum computer (Shor's polynomial time algorithm).
Step2: Call ComputeNum1 procedure on all the divisors of n
Step3: Sum the Num1 of all divisors of n and store it in Num2 variable
Step4: Return $Num2$

The algorithm to compute Primality Measure contains two main parts: One part computes Totient repeatedly. The other part computes the Num1 of all divisors of n.

It is well known that totient function can be computed in O(log n) steps on a Quantum Computer [4]. This can be seen by the equivalence of the factorization problem with the problem of finding Euler Totient Function. There exist polynomial time reductions from one to the other and we already know that Peter Shor has shown that factorization is polynomial time on a quantum computer.

Thus it just need to be proved that the repeated application of this O(log n) steps are also O(log n). This can be proved through the works of Pillai [2] who showed that the iterated totient function resolves to 1 in logarithmic number of steps. Thereby the computation of repeated Totients is also O(log n) time complexity.

Now to compute Num2, we can see that Polynomial Time Classical Algorithms do not exist for the same. To compute Num2, we need the list of divisors of n. This requires factorization of n. It is well known that there is currently no polynomial time classical algorithm for factorization. Hence, we rely on Peter Shor's Quantum Algorithm [2] for factorization. Peter Shor's Quantum Algorithm Factorization of a number can be done in O(polylog(n)). Thus, this part is also polynomial time.

Therefore, when we combine the two parts, we get a Polynomial Time Quantum Algorithm for computing the primality measure of a number.

III. CONCLUSION & FUTURE WORK

We propose that future work can be done creating a classical version of proposed Quantum Ramanil algorithm by removing the Quantum aspect of Ramanil Algorithm. The famous AKS Primality algorithm [1] is a purely classical polynomial time algorithm for primality. For the same one can rely on research related to the famous AKS Primality Algorithm which solves primality checking in polynomial time. We believe that practical and theoretical usages of this new proposed concept of primality measure will take place in future research work. One possible direction is in exploring how the Fermat's Little Theorem & Euler's theorem get modified through the new concept of Primality Measure with possible usages in Cryptography.

IV. ACKNOWLEDGEMENT

I have named this algorithm in loving & respectful memory of my father Late Sri Anil Kumar Saxena and my mother Smt Rama Saxena (Ramanil algorithm), whom I shall always remember with much gratitude and prayers to the Almighty. I also feel grateful to have attended a course on Computational Number Theory at IIT Kanpur during my M. Tech. I am also grateful to the referee of this paper.

REFERENCES

[1] M. Agrawal, N. Kayal, and N. Saxena, PRIMES is in P. Ann. of Math. (2), 160(2) (2004) 781–793.
[2] S. S. Pillai, On a function connected with φ(n), Bull. Amer. Math. Soc. 35(6) (1929) 837–841.
[3] Shor, Peter W., Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer, SIAM J. Comput., **26** (5) (1997) 1484–1509.
[4] G. L. Miller, 'Riemann's hypothesis and tests for primality', J. Comput. System Sci., 13 (1976), 300-317. MR0480295 (58:470a)